

A decorative graphic on the left side of the slide consisting of overlapping geometric shapes. It includes a blue parallelogram and a yellow parallelogram, both tilted at an angle, set against a dark blue background with subtle diagonal stripes.

saschavanessen@gmail.com

Python Course

Week 4:
Files and lists



Workshop Overview

1. Writing your first program
2. Making choices and reuse code
3. Loops and strings
4. Files and lists
5. Dictionaries and tuples

Acknowledgments: Structure of the workshop follows the book “Python for informatics” by Charles Severance. Several examples are from this book or the accompanying slides.



Assignments last week

- Count positives
- Count letters
- Guessing game

Recap





Recap past weeks

- Making choices
 - *if*, *elif*, *else* (test if something is True or False)
 - *try*, *except* (test if Python fails on something)
- Functions
 - *def*, arguments, *return*, values
- Loops
 - *while* (go on while a condition is True)
 - *for* (go through a list, range, string or other sequence)
 - *continue*, *break* (go to start of loop, break out of loop)
- Strings
 - *index* (count from 0 and get n^{th} character: `string[n]`)
 - *slice* (get part of the string from *n* to *m*: `string[n:m]`)



After this course? Examples

- <http://coursera.org>
 - Programming for everybody (repeat what you learned already with more examples)
 - Python data structures (idem)
 - Using Python to access web data
 - Using databases with Python
 - Interactive programming (e.g. games)
 - Raspberry Pi and Python (IoT)
- <http://edx.org>
 - MIT's Introduction to Computer Science and programming using Python
 - MIT's Computational thinking and data science
 - CS50's Introduction to Computer Science
- <https://www.codecademy.com>



Other languages

- Web: HTML5, Javascript, PHP
- Apps: Java, Swift, C, C++, C#
- Statistics/math: R, Matlab
- Electronics: C, Arduino
- Heavy calculations: C

Disclaimer: several languages can be used to do the same, but these are often used for these purposes and this is definitely not a complete list



Application areas

- Games
- Web applications
- Mobile applications
- Science, big data and/or math (e.g. life sciences, physics, finances)
- Cloud or high performance computing
- Computer graphics
- Etc etc etc!!



Example: data visualization

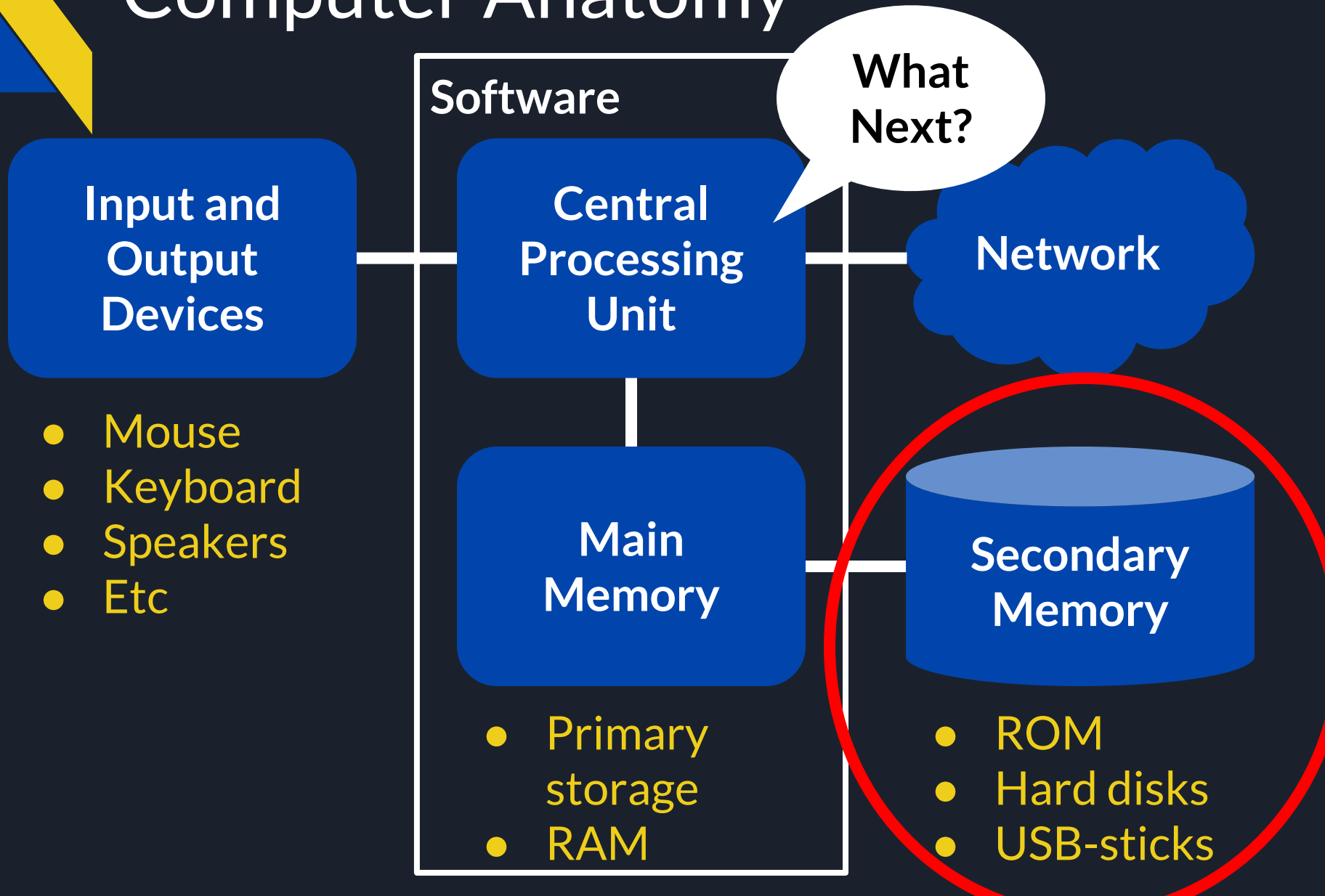
Visualize data with the Javascript D3 library

- <https://github.com/d3/d3/wiki/gallery>
 - <https://bl.ocks.org/larsenmtl/e3b8b7c2ca4787f77d78f58d41c3da91>
 - <https://www.jasondavies.com/wordcloud/>

Files

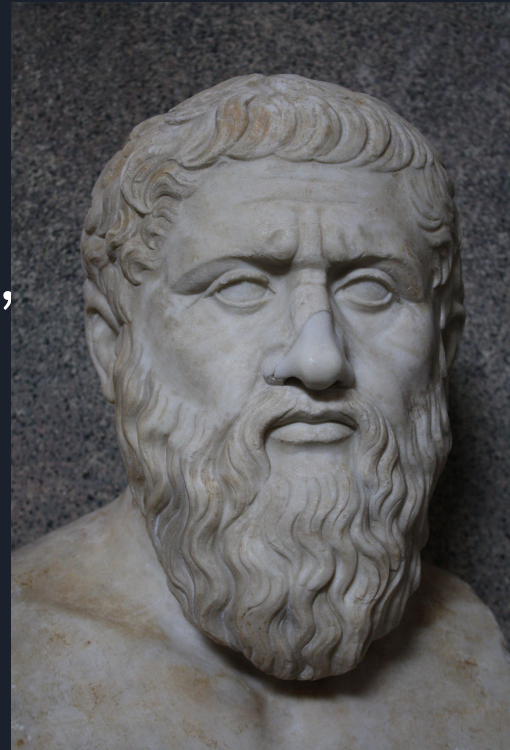


Computer Anatomy



Plato example file

- Illusion of logos
(just a random text from Plato)
- Persons of dialogue: Callicles,
Socrates, Chaerephon, Gorgias,
Polus
- Abbreviated with
Cal., Soc., Chaer., Gor., Pol.





Part of the text

Soc. How fortunate! will you ask him,
Chaerephon-?

Chaer. What shall I ask him?

Soc. Ask him who he is.

Chaer. What do you mean?



Read files

```
fp = open("plato.txt")
```

```
for line in fp:  
    print(line)
```

Why do you get extra empty lines
between the lines?

Answer: each line is read including
the return/newline at the end!



How to remove the newline?

```
fp = open("plato.txt")
```

```
for line in fp:
```

```
    line = line.strip()
```

```
    print(line)
```



Let the user choose a file

```
import sys
myfile = input("Enter filename: ")
try:
    fp = open(myfile)
except:
    sys.exit("cannot open file")

for line in fp:
    line = line.strip()
    print(line)
```




Via the commandline

```
import sys
```

Run it like this from the terminal:
python ask-user-for-file2.py plato.txt

```
if len(sys.argv) < 2:
```

```
    sys.exit("Usage: script.py file.txt")
```

```
myfile = sys.argv[1]
```

```
try:
```

```
    fp = open(myfile)
```

```
except:
```

```
    sys.exit("cannot open file")
```

```
...
```



Count lines in a file

```
import sys
myfile = "plato.txt"
try:
    fp = open(myfile, "r")
except:
    sys.exit("cannot open file")

count_lines = 0
for line in fp:
    count_lines = count_lines + 1
print("File contains", count_lines, "lines.")
```



Search in a file

```
...  
count_socrates = 0  
count_callicles = 0  
for line in fp:  
    line = line.strip()  
    if line.startswith("Soc."):  
        count_socrates = count_socrates + 1  
    elif line.startswith("Cal."):  
        count_callicles = count_callicles + 1  
print("Socrates:", count_socrates)  
print("Callicles", count_callicles)
```



Writing files (a)

```
fp = open("snoepjes.txt", "w")

for i in range(10):
    fp.write(str(i))
    fp.write(" Ik mag niet met
snoepjes gooien\n")

fp.close()
```



Writing files (b)

```
fp = open("snoepjes.txt", "w")
```

```
for i in range(10):  
    print(i, "Ik mag niet met  
snoepjes gooien", file=fp)
```

```
fp.close()
```

Lists





Lists and indices

```
>>> cijfers = [10,20,30,40,50,60]
>>> woorden = ["aap","noot","mies"]
>>> leeg = []

>>> print(woorden, cijfers, leeg)
>>> print(woorden[2])
>>> print(woorden[10])           # error!
```



Populate lists

```
>>> mylist = list(range(0,11,2))  
>>> print(mylist)
```

```
>>> zin = "Dit is een zin"  
>>> woorden = zin.split()  
>>> print(woorden)
```

Default: splits on a space or tab

For comma-seperated files use:

```
line.split(",")
```




Slices

```
>>> line = "a rose by any other name"  
>>> words = line.split()  
  
>>> print(words)  
>>> words[2:5]  
>>> words[0]  
>>> words[1][1]      # what do you expect?
```



List operations

```
>>> a = [1,6,9]
```

```
>>> b = [2,4,6]
```

```
>>> c = a + b
```

```
>>> c
```

```
>>> 3 * a
```

```
>>> d = [1,2,b]    # a list in a list!
```

```
>>> d
```

```
>>> len(d)    # what do you expect?
```



Mutability

- Strings are NOT mutable (last week)
- Lists are!

```
>>> c
```

```
>>> c[2] = 108
```

```
>>> c
```



Loop over lists

```
cheeses = ['cheddar', 'edam', 'gouda']
```

```
for cheese in cheeses:
```

```
    print(cheese)
```



Is element present in list?

```
>>> cheeses = ['cheddar', 'edam', 'gouda']
```

```
>>> 'edam' in cheeses
```

```
>>> 'brie' in cheeses
```



List methods (1)

```
>>> a = ["z", "o", "b"]
```

```
>>> b = ["e", "d", "c"]
```

- Add (append) to a list

```
>>> a.append("x")
```

- Extend list with other list

```
>>> b.extend(a)
```

- Sort list

```
>>> b.sort()
```



List methods (2)

- Pop: remove from list by index and return it

```
>>> b.pop(2)
```

- Pop default: remove last

```
>>> b.pop()
```

- Remove from list by element

```
>>> b.remove("x")
```



Apply functions to lists

```
>>> nums = [3,42,12,9,74,15]
```

```
>>> len(nums)
```

```
>>> max(nums)
```

```
>>> sum(nums)
```

```
>>> sum(nums)/len(nums)
```




List copies... or not?

```
>>> x = ['a', 'b', 'c']
```

```
>>> y = x
```

```
>>> z = ['a', 'b', 'c']
```

```
>>> x is y      # are they the same object?
```

```
>>> x is z      # are they the same object?
```

```
>>> x == z      # are they equal? (same values)
```

```
>>> y[0] = "iets"
```

```
>>> y
```

```
>>> x
```



Summary

- Files
 - Open, read, write
 - Parse lines from file
- Lists
 - A collection of words, letters, numbers and/or lists
 - List methods: append, pop, etc
 - Functions on lists: sum, len, etc
- Files and lists
 - Get words from a file or specific columns



Assignment! Hobbies

- Print the names of the persons in “hobbies.txt”
- Hints
 - Open file
 - “Split” the lines
 - Get the right column and print it



Assignment! Truth

- How many lines with the word “true” and how many with “false” in plato.txt?
- Hints
 - Open file
 - Loop over file lines
 - Make two variables to count “true” and “false”
 - Use the string method find:
https://www.tutorialspoint.com/python3/string_find.htm.
Notice the return value of this function!



Bonus! Guessing game

- Let the user think of a number between 1 and 1000
- The computer makes guesses
- The user gives hints: higher / lower (or h / l)

One solution: let the computer guess all the numbers between 1 and 1000... not very efficient.

How would you solve this as a concept?

How would you solve this with code?



To be continued!

- More practice:
 - Exercises in chapter 7 and 8 of the book
- Next week: Dictionaries and tuples
- See you next week!



Shortcuts

- Terminal

- Up previous commands
- Tab autocomplete
- cd .. one folder up

- Editor

- Ctrl-/ or Cmd-/ comment on/off
- Tab indent forward
- Shift-Tab indent backwards